Statistical Techniques in Robotics (16-831, S20) Lecture #10 (Monday, March 08)

Hard-SVM, Soft-SVM, OSGD

Lecturer: Kris Kitani *Scribes: Paritosh Mittal, Tanay Sharma*

1 Review

Last lecture introduced the concept of Gradient Descent and Stochastic Gradient Descent. It further explained how Online Gradient Descent is a special case of Online Mirror Descent (OMD). In this section, we will review these topics and develop tools that can be used to solve optimization problem for Max Margin Classifier (SVM). We will cover the (1) definition of convex functions, (2) Gradient Descent and (3) its links with OMD.

1.1 Convex Functions

A function $f : S \to \mathbb{R}$ is convex if for all $\mathbf{w}, \mathbf{v} \in S$

$$f(\alpha\mathbf{w} + (1-\alpha)\mathbf{v}) \leq \alpha f(\mathbf{w}) + (1-\alpha)f(\mathbf{v})$$

$$\text{for all } \alpha \in [0, 1]$$

Key properties of a convex smooth bounded function include (1) we can find the global minimum and (2) we can find it fast(er). We can formulate a convex optimization problem with a convex function and a convex solution space. However, the convex function needs to be convex-Lipschitz-bounded or convex-smooth-bounded.

1.2 Gradient Descent

Gradient Descent is an approach for minimizing differentiable (or non-differentiable in case of sub gradient descent) convex functions with regularization as added constraint. Consider a convex function f in figure 1, wherein we use Taylor series approximation of 1^{st} order, to lower bound $f(\mathbf{u})$

$$f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$$

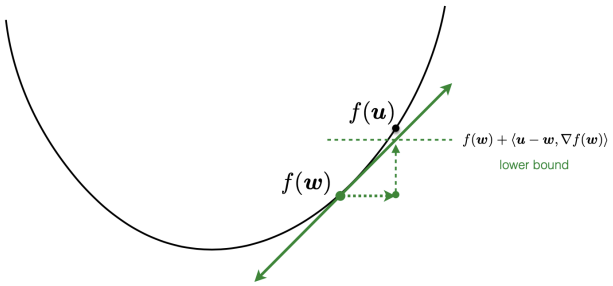

Figure 1: Taylor Series approximation of $f(\mathbf{u})$

If we directly select $\mathbf{u}$ that minimize f using only the previous inequality then we get solution at negative infinity. To avoid this we introduce a regularization term that constraints $\mathbf{w}$ to be close to $\mathbf{u}$. The final objective function for gradient descent hence looks like:

$$\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}^{(t)}||^2 + \eta(f(\mathbf{w}^{(t)}) + \langle \mathbf{w} - \mathbf{w}^{(t)}, \nabla f(\mathbf{w}^{(t)})\rangle \tag{1}$$

where η is a Lagrangian multiplier. Here we use gradient descent to solve the objective function in equation 1. If we observe closely then equation 1 is of the form of quadratic regularization + linear loss function. We know from past lectures that OMD solve objective functions with linear loss with quadratic regularization.

1.3 Online Gradient Descent(OGD): Special case of OMD

OMD is often used to solve optimization problems that contain a linear loss function $f(\mathbf{w})$ and a quadratic regularization $\psi(\mathbf{w})$. In case of OGD, we can write them as:

$$\psi(\mathbf{w}) = \frac{1}{2\eta}||\mathbf{w}||_2^2 \text{ and } f(\mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\theta}\rangle$$

Where $\boldsymbol{\theta}$ is the parameter for dual space. We can see that $f(\mathbf{w})$ is linear and $\psi(\mathbf{w})$ is quadratic for OGD. Further when we solve for optimal parameter to minimize the convex function, we notice

$$w_n = \eta\theta$$

Hence the mirror function for OGD is $g(\boldsymbol{\theta}) = \eta\boldsymbol{\theta}$. Therefore OGD is OMD with a linear loss & quadratic regularization (or quadratic loss with linear constraints). We will soon notice that optimization problem for max-margin classifier (SVM) is of similar form and hence OGD can be used to solve it.

2 Summary

2.1 Hyperplanes

A hyperplane of an n-dimensional vector space is geometrically defined as a subspace of dimension n-1 [3]. In simple terms, a hyperplane of 2D vector space is a line and that of a 3D vector space is a 2D plane. Mathematically, hyperplane is defined as

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \text{ where } \mathbf{w} \in \mathcal{R}^n \text{ for n dimensional space}$$

Key observations for hyperplane include

- **We can choose any normalization of w**: Equality in hyperplanes are to scale, i.e. if we multiply an equation of hyperplane with non-zero factor λ then we get the same hyperplane

$$w_1x_1 + w_2x_2 + b = 0 \text{ and } \lambda(w_1x_1 + w_2x_2 + b) = 0 \text{ correspond to same hyperplanes}$$

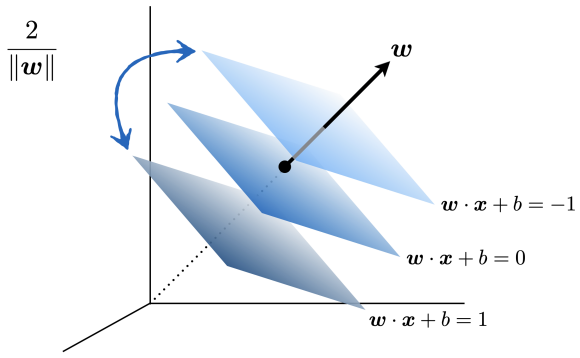

Figure 2: Distance between parallel hyperplanes

- **Distance of hyperplane from origin**: The perpendicular distance of a hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ from the origin of n-dimensional vector space is $b/||\mathbf{w}||$

$$\text{if we scale: } \mathbf{w} \cdot \mathbf{x} + b = 0 \text{ with } \frac{1}{||\mathbf{w}||}$$

$$\text{transformed to normal form: } x\cos\theta + y\sin\theta = \rho$$

$$\text{we get distance from origin: } \rho = \frac{b}{||\mathbf{w}||}$$

- **Distance between parallel hyperplanes**: Consider two parallel hyperplanes defined by (a) $\mathbf{w} \cdot \mathbf{x} + b = 1$ and (b) $\mathbf{w} \cdot \mathbf{x} + b = -1$

$$\text{distance of hyperplane (a) from origin: } \frac{b-1}{||\mathbf{w}||}$$

$$\text{distance of hyperplane (b) from origin: } \frac{b+1}{||\mathbf{w}||}$$

$$\text{perpendicular distance between these hyperplanes: } \frac{2}{||\mathbf{w}||}$$

Note that because we can choose any normalization of $\mathbf{w}$ for the hyperplanes, we can always write the distance between two parallel hyperplanes as $2/||\mathbf{w}||$. This is hence used as convention (fig. 2).

2.2 Support Vector Machine (Max Margin Classifier)

Consider a case of binary linearly separable data ($\mathcal{D}$), the task at hand is to learn a linear decision boundary that perfectly classifies this data. We know from our knowledge of Perceptron, that there are infinite such hyperplanes possible for linearly separable data. However, Max Margin Classifier learns a decision boundary (surface) with maximum margin (γ).

Definition 1. ***Margin***(γ) *of a set of points in* $\mathcal{D}$ *is defined as the distance from the decision surface to the closest point in* $\mathcal{D}$*.*
Refer to figure 3 for better visualization. Without loss of generality, let $\mathbf{w} \cdot \mathbf{x} + b = 0$ be the

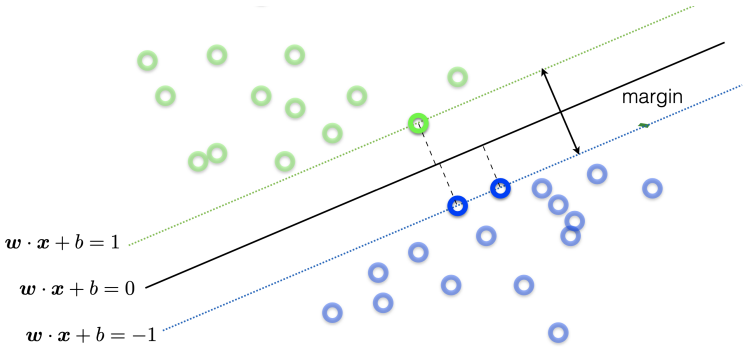

Figure 3: Highlights the max margin classifier along with the margin hyperplanes

equation for max-margin classifier such that parallel hyperplanes $\mathbf{w} \cdot \mathbf{x} + b = 1$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$ pass through the nearest data points to classifier hyperplane. The distance between these planes correspond to the margin of the classifier. Hence we know that margin for linearly separable data with decision boundary $\mathbf{w} \cdot \mathbf{x} + b = 0$ is

$$\frac{2}{||\mathbf{w}||}$$

Hence the objective of the max-margin classifier is

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \quad \text{subject to}$$
$$\mathbf{w} \cdot \mathbf{x} + b \begin{cases} \geq +1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \quad \text{for } x_i \in \mathcal{D}$$

Equivalently, we can simplify the problem as

$$\min_{\mathbf{w}} ||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x_i} + \mathbf{b} \geq 1) \text{ for } i = 1, \cdots, N$$

We identify $\min_{\mathbf{w}} ||\mathbf{w}||^2$ as the objective function and $y_i(\mathbf{w} \cdot \mathbf{x_i} + \mathbf{b}) \geq 1$ as the "Hard" constraints. There exists a unique solution for this problem and we can use tools of convex quadratic programming (QP) problem to solve it.
We make a major assumption here that data ($\mathcal{D}$) is linearly separable, this is generally not the case for most real life scenarios. To counter this problem there is a popular variant of SVMs termed as "Soft" SVMs which allow for some 'slack'.

2.3 Soft Margin SVM

Intuition

We saw in the previous section that SVM has to satisfy the hard constraints to converge and find a decision boundary for a separable data. But it is quite possible in real world data to have a few samples which lie very close to the samples of the opposite label as shown in Fig 4. even while maintaining data separability. In such scenarios, a hard SVM will try to find the maximum margin from the nearest sample and the decision boundary will look like as given in Fig 4. This brings us

to the intuition that if we relax our constraints a bit and allow some misclassification, we can find a better margin. This is the notion of soft margin.

We should allow for some misclassification if we can get more robust classification

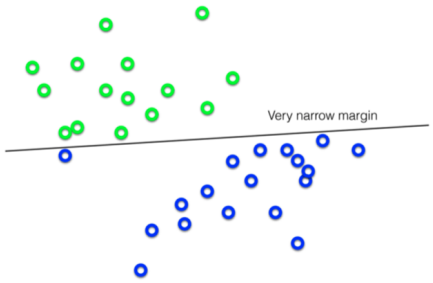

Figure 4: Hard Margin

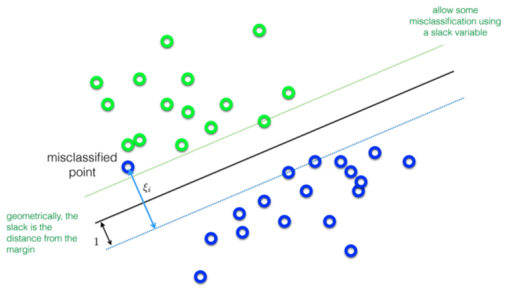

Figure 5: Soft Margin with ξ

Slack Variable

To allow this tradeoff between the margin and the mistakes, a slack variable is introduced in the constraint. The slack variable ξ_i is defined as the distance of the point from the margin as shown in Fig 5. We can control the extent of mistakes allowed by our algorithm by this parameter ξ_i. After including this variable, our hard constraint tranforms into a soft constraint that allows some mis-classification as,

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$
$$\text{where } \xi_i \geq 0$$

Objective Function

Following from our previous objective function in hard constraint case, now we need to minimize the sum of ξ_i along with the weight vector. Hence our objective function becomes,

$$min_{w,\xi}(\|w\|^2 + C\sum_i \xi_i)$$
$$\text{subject to}$$
$$y_i(w^T x_i + b) \geq 1 - \xi_i$$
$$\text{for i} = 1 \text{ to N}$$

We can see from the objective function that

1. If C tends to infinity, we end up giving a lot of weight to minimize the mistakes and thus the problem converts to hard margin case.
2. If we reduce C to a very small value , the optimization function ends up reducing the weight vector towards 0

Hence, C can be chosen as a value in between for instance 10. In this case it will be termed as a soft margin.

Now to incorporate linear constraints into the objective function, we replace the value of ξ_i from the inequality and put it in the objective function.

$$min_w(\frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_{i=0}^{M} 1 - y_i w^T x_i) \tag{2}$$

where, λ is an inverse multiplier at the place of C
M = number of training samples

In the equation 2 we have included ξ of all the points to minimize without any added constraint. This creates problem by making the second term negative when more number of points are correctly classified (easy examples). This overpowers the loss from wrongly identified points and thus fail to generalize.

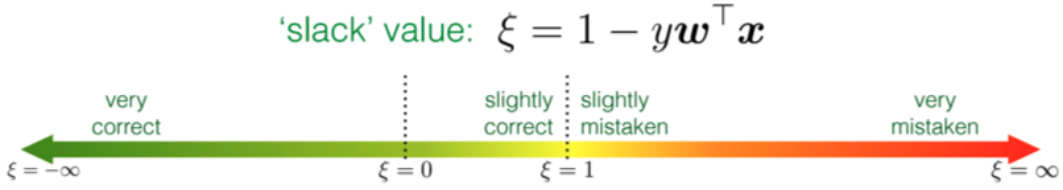

Figure 6: Spectrum of ξ and its interpretation

Since we do not gain by minimizing the loss on easy examples, we can avoid them all together and concentrate on the region where our point is slightly correct, slightly mistaken or very mistaken. As can be seen in Fig 6, such region corresponds to $\boldsymbol{\xi > 0}$. This observation motivates us to clip the values of $\xi < 0$ in the objective function. And thus we can replace the second part of the objective function with a **Hinge Loss** which is defined as,

$$HingeLoss = max(0, 1 - y_i w^T x_i) \tag{3}$$

Substituting equation 3 in 2 we get our final objective function as,

$$min_w(\quad \underbrace{\frac{\lambda}{2}\|w\|^2}_{\textbf{regularization}} \quad + \underbrace{\frac{1}{M}\sum_{i=0}^{M} max(0, 1 - y_i w^T x_i))}_{\textbf{loss function}} \tag{4}$$

As we can see in equation 4, we have finally converted our objective function to a familiar function we saw in Online Gradient Descent. ***It is a linear function with a quadratic regularization.***

Properties of this function

1. It is a convex function
2. It is not differentiable because of Hinge Loss

To solve such convex optimization problems where the function is not differentiable we can use a variant of Gradient Descent called **Online Sub-Gradient Descent**

2.4 Sub Gradients

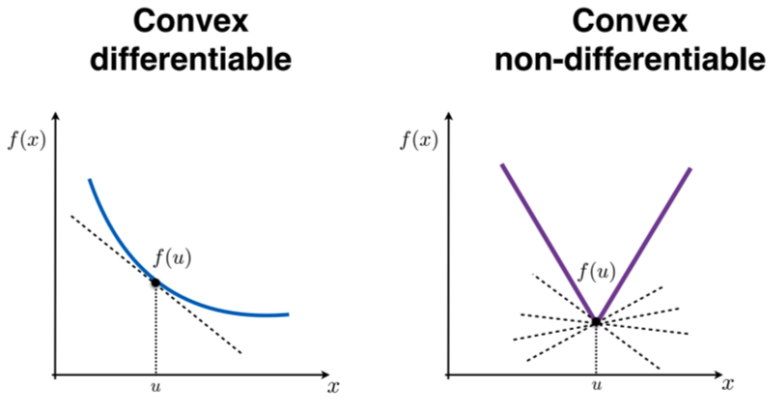

Figure 7: Gradient and Sub-Gradient

Definition 2 (Sub-Gradients[1])**.** *The subgradient of a convex function f at w_0 is formally defined as all vectors v such that for any other point w,*

$$f(w) - f(w_0) \geq v(w - w_0)$$

If f is differentiable at w_0, then the subgradient contains only one vector which is the gradient $\delta f(w_0)$. However, when f is not differentiable, there may be many different values for v that satisfy this inequality as shown in Fig 7.

For our problem statement, Hinge Loss is not differentiable at 0 and thus we need to use sub-gradients at that location. Since, there can be many sub-gradients, we can choose 2 sub-gradients based on the direction of the function. One for the part of the function greater than 0 and one for the part less than 0.

$$z_m = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 1, \\ -y_i x_i & \text{otherwise} \end{cases} \tag{5}$$

Since we have all the tools required for optimization, we can solve this by using Online Sub-Gradient Descent. This algorithm works same as Online Gradient Descent as explained before but uses sub-gradients at the place of gradients. Now we can write the complete algorithm of Soft SVM in the format of Online Mirror Descent(OMD) as this is also a special case of OMD.

Algorithm 1 SoftSVM(λ)

$\theta^{(1)} \leftarrow \{0 \in R^N\}$ ▷ Weight initialization
for $t = 1, \cdots, T$ **do**
  $y_d, \boldsymbol{x}_d \sim D$ ▷ Receive sample from environment
  $\theta^{(t)} = \theta^{(t-1)} + y_d x_d . \mathbb{1}[y_d(w^{(t)}.x_d) < 1]$ ▷ Dual parameter update
  $w^{(t+1)} \leftarrow \frac{1}{\lambda+1}\theta^{(t)}$ ▷ Mirror projection
end for
$w = \frac{1}{T}\sum_t w^t$

2.5 Comparison of Soft SVM with Perceptron Algorithm

Algorithm 2 Perceptron Algorithm

$\theta^{(1)} \leftarrow \{0 \in R^N\}$ ▷ Weight initialization
for $t = 1, \cdots, T$ **do**
  RECEIVE$(y^{(t)}, \boldsymbol{x}^{(t)})$ ▷ Receive sample from environment
  $\theta^{(t)} = \theta^{(t-1)} + y^{(t)}x^{(t)} . \mathbb{1}[y^{(t)}\langle w^{(t)}.x^{(t)}\rangle < 0]$ ▷ Dual parameter update
  $w^{(t+1)} \leftarrow \theta^{(t)}$
end for

As we can see in the above algorithms

1. Both the algorithms have similar dual parameter update. The only difference is that SVM uses soft margin while perceptron uses no margin.

2. Both use similar mirror function.

3 Appendix

3.1 Implementing SVM on linearly inseparable data

Real life data is often linearly inseparable. However, **Cover's Theorem** states that "pattern-classification problem cast in a high dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space" [2]. This signifies that we can apply non-linear transformations on low-dimensional data in order to transform them to high dimensional vector space where they are more likely to be linearly separable. Consider the case of 1D data in figure 8 which is not separable by a linear function.

Figure 8: Linearly inseparable data [2]

We can apply a non-linear function $\phi(\mathbf{x}) = (\mathbf{x}, \mathbf{x^2})$ to lift it in 2D space as illustrated in figure 9

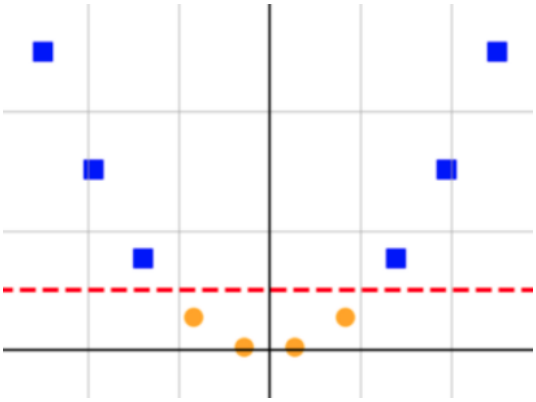

Figure 9: Same Data made linearly separable[2]

We can see that in this new high dimensional space, the projected data becomes linearly separable. We can then use max-margin classifier to learn a hyperplane that can separate the data points. **Kernel Tricks** is a very popular and efficient way of making a linearly inseparable data, separable. The SVM optimization function depends only on the dot product ($\mathbf{x_i^T x_j}$) of data points and hence if we know a function $\mathbf{K}$ such that

$$\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$$

then we do not explicitly need to calculate the mapping function (ϕ) for increasing the dimension of input space. $\mathbf{K}$ is called the kernel function and calculating $\mathbf{K}$ is computationally efficient.

References

[1] D. Sontag and Y. Halpern. Gradient, subgradient and how they may affect your grade(ient). 2016.

[2] O. Veksler. cs434a/541a :pattern recognition, lecture 11.

[3] Weisstein and W. Eric. "hyperplane." from mathworld–a wolfram web resource.